EECS 349 (Machine Learning) Homework 1

WHAT TO HAND IN

1. A PDF document containing answers to the homework questions. Show your reasoning in your answers.
2. The source code for the program you write (in Python 2.7 using SciPy, if needed).

HOW TO HAND IT IN

1. Compress all of the files specified into a .zip file.
2. Name the file in the following manner, firstname_lastname_hw1.zip. For example, Bryan_Pardo_hw1.zip.
3. Submit this .zip file via Canvas

DUE DATE: See course calendar.

Problem 1 (1 point) :

Given the following:

Instances ***X***: cars described by following attributes

Origin: {Japan, USA, Korea, Germany}
Manufacturer: {Honda, Chrysler, Toyota}
Color: {Blue, Green, Red, White}
Decade: {1970, 1980, 1990, 2000}
Type: {Economy, Sports, SUV}

Target function **c**:

JapaneseEconomyCar → {0,1}

Hypothesis set ***H***

Conjunctions of literals, ? and 0, as described in Section 2.2 (page 21)

Training examples D

Origin	**Manufacturer**	**Color**	**Decade**	**Type**	**Classification**
Japan	Honda	Blue	1980	Economy	**1**
Japan	Toyota	Green	1970	Sports	**0**
Japan	Toyota	Blue	1990	Economy	**1**
USA	Chrysler	Red	1980	Economy	**0**
Japan	Honda	White	1980	Economy	**1**

Perform the Candidate-Elimination learning algorithm on the training examples. Show the S and G sets at every step.

Problem 2 (3 points):

A) (1 point) Given 2 people on Facebook, define a distance metric between them. Prove that it is a metric. This means you have to identify features you will use to distinguish people's Facebook pages, define a way to measure the differences between them, given the features…and then use a math proof to show this measurement is a metric.

B) (1 point) Assume each point in your space represents a person encoded as 3-element vector: <height in feet, weight in kilograms, number of hairs on their head>. Give an expected range of values for each of these elements. You don't have information on age, but the goal is to cluster people into 4 groups "child, teen, middle-aged, elderly". Does it make sense to put people in a Euclidean 3-space where we treat values for all three elements equally? Why or why not? How would you design a metric to cluster people appropriately? Explain your reasoning.

C) (1 point) A strand of DNA can be encoded as a sequence of four bases—adenine, guanine, cytosine, and thymine. Consider the metric for strings defined in "the String to String Correction Problem" (a paper available on the course calendar). Explain how this could be used to determine the distance between two strands of DNA. Does the metric need to be adapted in any way, or could it be used, essentially as-is? Back up your statement.

Problem 3 (2 points):

A typical machine learning task is classification: give example $\boldsymbol{x}$ the right class label $\boldsymbol{l}$. We assume there is a function $\boldsymbol{f(x)}$ whose domain is the set of examples $\boldsymbol{X}$ and whose range is the set of possible labels $\boldsymbol{L}$. The task is to learn that function. A learner does this by building and testing hypotheses. A hypothesis $\boldsymbol{h(x)}$ is a function whose domain is $\boldsymbol{X}$ and whose range is $\boldsymbol{L}$. In supervised learning, the selection of hypotheses is guided by the use of a finite training set, $\boldsymbol{T}$, containing examples labeled by the function we are trying to learn.

Definition: Two functions $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ are *distinguishable*, given a set of examples $\boldsymbol{E}$, if they differ in their labeling of at least one of the examples in $\boldsymbol{E}$.

The goal of learning is to find a hypothesis $\boldsymbol{h}$ that is indistinguishable from $\boldsymbol{f}$.

Definition: A set of hypotheses is distinguishable iff ALL pairs of hypotheses in the set are distinguishable given some set of examples. Call $\boldsymbol{D_E}$ a largest set of distinguishable hypotheses, given example set $\boldsymbol{E}$.

A) (1 point) Let the size of the labeling domain be a fixed value drawn from the counting numbers (i.e. $|L| \in \{1,2,3...\}$). What formula governs the size of $\boldsymbol{D_T}$, given the training set $\boldsymbol{T}$ and $|\boldsymbol{L}|$? Explain your reasoning.

Now, Let the label set $\boldsymbol{L}$ be $\{0,1\}$. Let the size of the training set $|\boldsymbol{T}| = 100$ and the size of the set of all possible examples $|\mathbf{X}| = 200$. Assume a learner able to consider a maximal set of distinguishable hypotheses $\boldsymbol{D_T}$, given $\boldsymbol{T}$.

B) (1/2 point) What is the probability that the learner will find a hypothesis $\boldsymbol{h}$ that is indistinguishable from the target function $\boldsymbol{f}$, given $\boldsymbol{T}$? Explain your reasoning. State your assumptions.

C) (1/2 point) Assume the learner HAS found a hypothesis $\boldsymbol{h}$ that is indistinguishable from the target function $\boldsymbol{f}$, given $\boldsymbol{T}$. What is the probability that hypothesis $\boldsymbol{h}$ that is indistinguishable from the target function $\boldsymbol{f}$, given $\mathbf{X}$? Explain your reasoning. State your assumptions.

Problem 4 (2 points):

A) (1 point) Calculate the decision tree that would be learned from the training examples in the "Japanese Economy Car" concept from Problem 1 by running ID3 (see the Mitchell textbook). Show information gain for each attribute at each level of the tree. Feel free to write code to do this. This code could be reused for problem 6.

B) (1 point) Write out the logical function encoded in the decision tree for "Japanese Economy Car." How well does this function categorize the data used to build the tree? How well do you feel this captures the concept of "Japanese Economy Car"? What about the approach or the data caused this result?

EECS 349 (Machine Learning) Homework 1

Problem 5 (2 points):

Weka is a collection of machine learning algorithms for data mining tasks available here (http://www.cs.waikato.ac.nz/~ml/index.html). The algorithms can either be applied directly to a dataset or called from your own Java code.

Download the latest stable version of Weka. **Using the Weka Explorer**, open the IvyLeague.txt data set (described in problem 6). When you try to open it, Weka will complain and ask you to specify a data loader – use the default CSV loader which it suggests to you. Note that there are some tools for visualizing the data in the file.

A) (1/2 point) Run the ID3 Weka decision tree classifier algorithm (named "classifiers.trees.Id3" within Weka) on the IvyLeague.txt data set (see Problem 6). Show us the output, using 10-fold cross-validation. Show us the textual representation of the decision tree that was created, as well as the "Confusion Matrix". Briefly state what the numbers in the Confusion Matrix mean. NOTE: This might mean you will have to look up what a confusion matrix is. Try the web.

B) (1/2 point) Run the "classifiers.trees.J48" classifier (which is an implementation of the C4.5 decision tree algorithm) on the IvyLeague.txt dataset, again with 10-fold cross-validation. Note that the default settings for J48 ("-C 0.25 -M 2") will cause some pruning to occur. Again show us the textual decision tree that was created, and the Confusion Matrix. Did the C4.5 algorithm with pruning outperform the straightforward ID3 algorithm in this case? In a sentence or two, offer a hypothesis explaining your results.

C) (1/2 point) Compare the performance of an ID3 decision tree learner on the IvyLeague.txt and MajorityRule.txt example files. Is there a difference in performance on these two data sets? If so, explain what about the interaction between the learning bias of the ID3 algorithm and the structure of the concepts in the data sets caused this difference.

D) (1/2 point) Reduced error pruning works as follows: Starting at the leaves, each node is replaced with its most probable class. If the prediction accuracy on a validation set is not affected then the change is kept. Would you expect Reduced Error Pruning to have a large positive effect on the performance of an ID3-generated decision tree when measured on the examples MajorityRule.txt? Why or why not?

Problem 6 (5 points): Create a Decision Tree classifier that builds a tree using the ID3 algorithm described in the textbook and in class.

Now that you've tried out a decision tree package, you are going to code up a decision tree, yourself. Recall that decision tree induction is a machine learning approach to approximating a target concept function *f*, given a set of examples. An *example* is a tuple $<x_1, x_2, \ldots, x_n, f(x_1, x_2, \ldots, x_n)>$ consisting of values for the *n* inputs to the function *f* and the output of *f*, given those values.

For this problem, you will construct a binary decision tree learner.

PROVIDED FILES

You have been provided with two example input files for the learner.

IvyLeague.txt – a file containing examples of target concept "people who will be accepted to an Ivy League university"

MajorityRule.txt – a file containing examples of target concept "over half of them are 'true'"

An input file is expected to be an ASCII file, where the first line is a tab-delimited list of *j* attribute names, followed by the word "CLASS". Each subsequent line consists of a fixed number of tab-delimited values. Each value must be the string **'true'** or the string **'false'** and there must be one value for each attribute in the first line of the file, plus an additional Boolean value (at the end of each line) that specifies how the target concept function would classify the example defined by these attribute values. The task for a machine learner is to learn how to categorize examples, using only the values specified for the attributes, so that the machine's categorization matches the categorization specified in the file. What follows is an example 3-line input file for a target concept 'People accepted to an Ivy League School'

EECS 349 (Machine Learning) Homework 1

```
GoodGrades GoodLetters GoodSAT IsRich Scholarship ParentAlum SchoolActivities CLASS
true  true  true  true  true  true  true  true

true  true  true  false  true  true  false false
```

YOUR PROGRAM

Your program must be written in Python 2.7 and run in **the test environment** without alteration. We are not responsible for debugging code written in other languages. We are not responsible for linking to any 3rd party input/output libraries, except for SciPy. The core decision tree algorithm code must be written by you. Your source code must be well commented so that can be easily understood.

Your folder MUST contain a file named "decisiontree.py" that can be called according to the following usage spec:

```
Usage: python decisiontree.py <inputFileName> <trainingSetSize> numberOfTrials> <verbose>
```

Note: You can read our input files with ease by using the module csv, which comes with Python 2.7. The items in the files are tab-delimited, with each new line beginning a new row. You could read each row as a dictionary, so that the element looks like, for example the first row could be read as follows:

```
{'GoodGrades': True, 'GoodLetters': True, 'GoodSAT': True, 'IsRich':
True, 'HasScholarship': True, 'ParentAlum': True, 'SchoolActivities':
True}
```

Note: Use string comparisons to change the string 'true' to the python boolean True. The format above will probably make it easier to write up the code.

What the Parameters Do

```
inputFileName    - the fully specified path to the input file. Note that windows
                   pathnames may require double backslash '\\'
trainingSetSize - an integer specifying the number of examples from the input
                   file that will be used to train the system
numberOfTrials  - an integer specifying how many times a decision tree will be built
                   from a randomly selected subset of the training examples.
verbose          - a string that must be either '1' or '0'
                   If verbose is '1' the output will include the training and test
                   sets. Else the output will only contain a description of the tree
                   structure and the results for the trials.
```

What the Program Must Do

When run, your program must perform the following steps.

1) Read in the specified text file containing the examples.
2) Divide the set of examples into a training set and a testing set by randomly selecting the number of examples for the training set specified in the command-line input <trainingSetSize>. Use the remainder for the testing set.
3) Estimate the expected prior probability of TRUE and FALSE classifications, based on the examples in the training set.
4) Construct a decision tree, based on the training set, using the approach described in Chapter 3 of Mitchell's Machine Learning.
5) Classify the examples in the testing set using the decision tree built in step 4.
6) Classify the examples in the testing set using just the prior probabilities from step 3.
7) Determine the proportion of correct classifications made in steps 5 and 6 by comparing the classifications to the correct answers.
8) Steps 2 through 7 constitute a *trial*. Repeat steps 2 through 7 until the number of trials is equal to the value specified in the command-line input <number of trials>.

9) Print the results for each trial to an output file to the command line (standard output). The format of the output is specified in the following section.

OUTPUT FORMAT

Each run of your decision tree program should create an output on the command line that contains the following information:

- The input file name
- The training set size
- The testing set size
- The number of trials
- The mean classification performance of the decision tree on the testing sets
- The mean classification performance on the testing sets achieved by using the probability of "true," as derived from the training sets.

For each trial your program should output

- The number of the trial
- The proportion of correct classifications of the test set returned by the decision tree
- The proportion of correct classifications returned by applying prior probability (based on probabilities derived from the training set) to classify the test set
- The structure of the decision tree built from the training set

When VERBOSE = 1 you must provide the following information for each trial:

- The set of examples in the training set
- The set of examples in the testing set
- The classification returned by the decision tree for each member of the testing set
- The classification returned by applying prior probability to each member of the testing set.

We will not require that a particular layout be used. That said, if we have ANY problem finding the information specified above in your output file, you WILL lose points. It is up to you to make your output format CLEAR.

EECS 349 (Machine Learning) Homework 1

What follows is an example output for a non-verbose run with two trials in a format that would be completely acceptable.

```
TRIAL NUMBER: 0
--------------------

DECISION TREE STRUCTURE:
parent: root  attribute: IsRich  trueChild:GoodLetters  falseChild:HasScholarship
parent: IsRich  attribute: GoodLetters  trueChild:leaf  falseChild:GoodGrades
parent: GoodLetters -
parent: GoodLetters  attribute: GoodGrades  trueChild:leaf  falseChild:leaf
parent: GoodGrades -
parent: GoodGrades -
parent: IsRich  attribute: HasScholarship  trueChild:GoodLetters  falseChild:leaf
parent: HasScholarship  attribute: GoodLetters  trueChild:GoodSAT  falseChild:leaf
parent: GoodLetters  attribute: GoodSAT  trueChild:leaf  falseChild:leaf
parent: GoodSAT -
parent: GoodSAT -
parent: GoodLetters -
parent: HasScholarship -

       Percent of test cases correctly classified by a decision tree built with ID3 = 88%
       Percent of test cases correctly classified by using prior probabilities from the
training set = 45%

TRIAL NUMBER: 1
--------------------

DECISION TREE STRUCTURE:
parent: root  attribute: IsRich  trueChild:GoodSAT  falseChild:HasScholarship
parent: IsRich  attribute: GoodSAT  trueChild:leaf  falseChild:GoodGrades
parent: GoodSAT -
parent: GoodSAT  attribute: GoodGrades  trueChild:GoodLetters  falseChild:leaf
parent: GoodGrades  attribute: GoodLetters  trueChild:leaf  falseChild:SchoolActivities
parent: GoodLetters -
parent: GoodLetters  attribute: SchoolActivities  trueChild:leaf  falseChild:leaf
parent: SchoolActivities -
parent: SchoolActivities -
parent: GoodGrades -
parent: IsRich  attribute: HasScholarship  trueChild:leaf  falseChild:leaf
parent: HasScholarship -
parent: HasScholarship -

       Percent of test cases correctly classified by a decision tree built with ID3 = 76%
       Percent of test cases correctly classified by using prior probabilities from the
training set = 43%

example file used = IvyLeague.txt
number of trials = 2
training set size for each trial = 20
testing set size for each trial = 42
mean performance of decision tree over all trials = 82% correct classification
mean performance of using prior probability derived from the training set = 44% correct
classification
```